MetaSymploit: Day-One Defense Against Script-based Attacks with Security-Enhanced Symbolic Analysis

Ruowen Wang[1], Peng Ning[1], Tao Xie[2], Quan Chen[1]
[1]North Carolina State University
[2]University of Illinois at Urbana–Champaign

Computer Science
NC STATE UNIVERSITY

Background

Computer Science
NC STATE UNIVERSITY

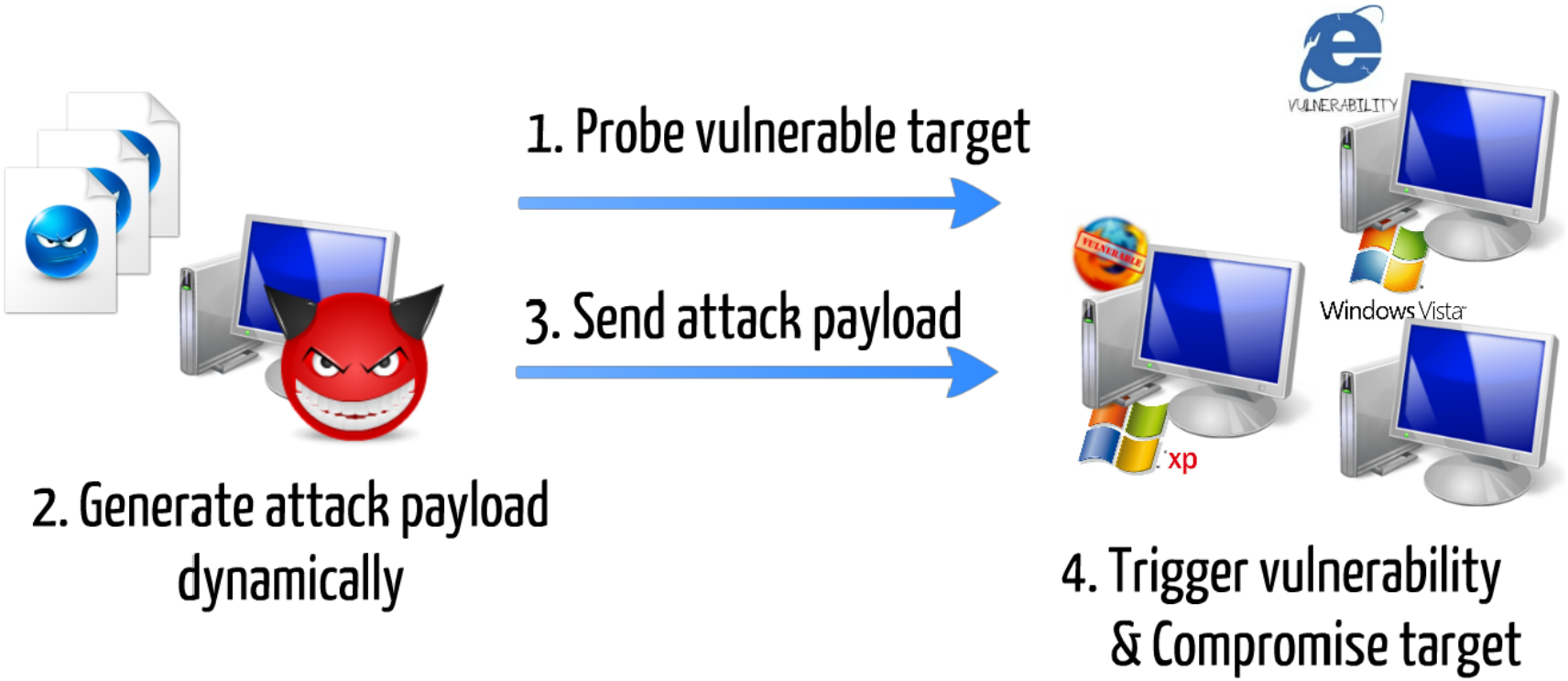
Script-based Attack Framework

- Written in scripting languages (e.g., Ruby, Python)
- Run on the attacker side, generate specific attack payloads to exploit various vulnerable targets
- All-in-one framework with built-in components providing rich APIs
- Support quick development, making the cost of attack is much lower than the cost of defense
- http://sectools.org/tag/sploits/

One Example: Metasploit

- Ruby-based penetration framework
- 1000+ (keep increasing) attack scripts
- Target all popular OS platforms
- One attack script can generate different payloads
- Script kiddie/Bot friendly

Metasploit Exploit Mechanism



Running Example

```
1 def exploit
2   connect()
3   preamble = "\x00\x4d\x00\x03\x00\x01"
4   version = probe_ver()
5   if version == 5
6     payload = prep_ark5()
7   else
8     payload = prep_ark4()
9   end
10  preamble << payload.length
11  sock.put(preamble) # Required by protocol
12  sock.get_once()
13  sock.put(payload) # Send attack payload
14  sock.get_once()
15  ... # vulnerability triggered
16 end
17 def prep_ark5()
18  payload = shellcode()
19  payload << rand_alpha(1167 - payload.length)
20  payload << "\xe98" + [-1172].pack("V")
21  payload << "\xeb\xf9"
22  payload << get_target_ret(5) # Tar_Ver: 5
23  payload << rand_alpha(4096 - payload.length)
24  return payload
25 end
```

1. Probing Target
   Port scanning, Fingerprinting, etc.
2. Compose Attack Payload
   Include shellcode, padding, target-specific vul bytes, etc.
3. Send Payload
   Trigger vulnerability
4. Post Exploit
   Wait shellcode to be executed, backdoor channel created, etc.

Computer Science
NC STATE UNIVERSITY

Malicious Use of Metasploit

The exploits
by Rapid 7 t
their networ

The exploits
catastrophic
critical infras
these device
physical w

ComputerWeekly.com

News | IT Management | Industry Sectors | Technology Topics | Blogs | Multimedia | Vendor Content | Jobs

Evaluating and adopting PaaS for your environment – our experts separate vendor-hype from actual benefits.

Home > Topics > IT security > Hackers and cybercrime prevention > Java zero-day vulnerability hits Metasploit and Blackhole

NEWS

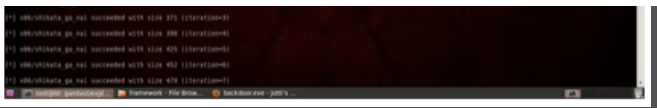
Java zero-day vulnerability hits Metasploit and Blackhole

Warwick Ashford
Thursday 30 August 2012
09:30

Share | +1 1 | Tweet 21

The latest Java zero-day vulnerability is already available to users of the Metasploit tool and Blackhole exploit kit, say security researchers.

The Java vulnerability allows attackers to use a custom web page to force systems to download and run malware that does not have to be coded in Java.



owned
test
ack.

tial
rators of
urity of

Computer Science
NC STATE UNIVERSITY

Motivation

- An effective defense is needed against these attack scripts
  - Catch up the release speed of new attack scripts
  - Provide quick defense using existing IDS
  - Prevent public exploit resource misuse

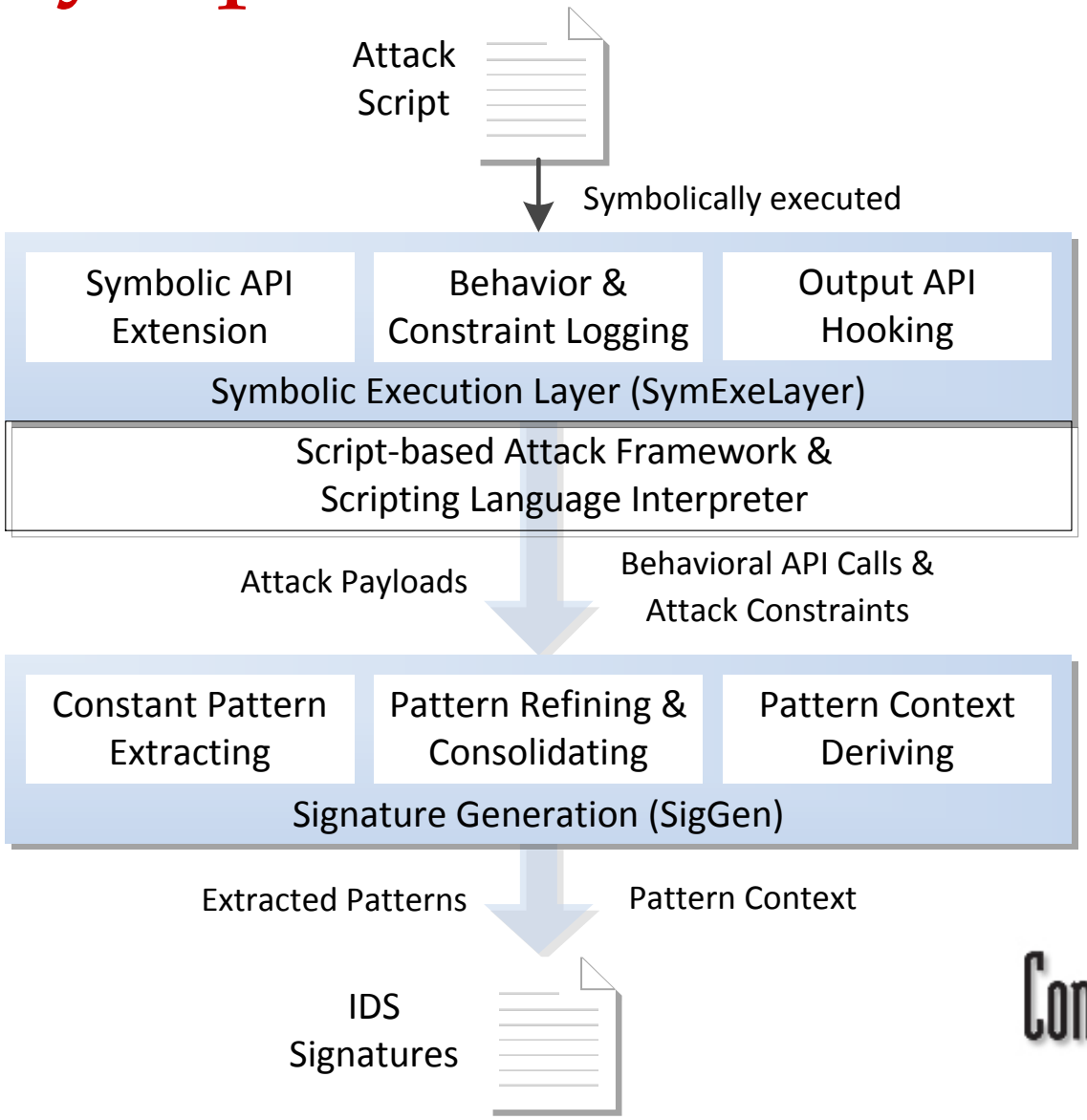
Our Work: MetaSymploit

- The First system of
  - Fast Attack Script Analysis
  - Automatic IDS Signature Generation
  - Using Security-enhanced Symbolic Analysis

MetaSymploit

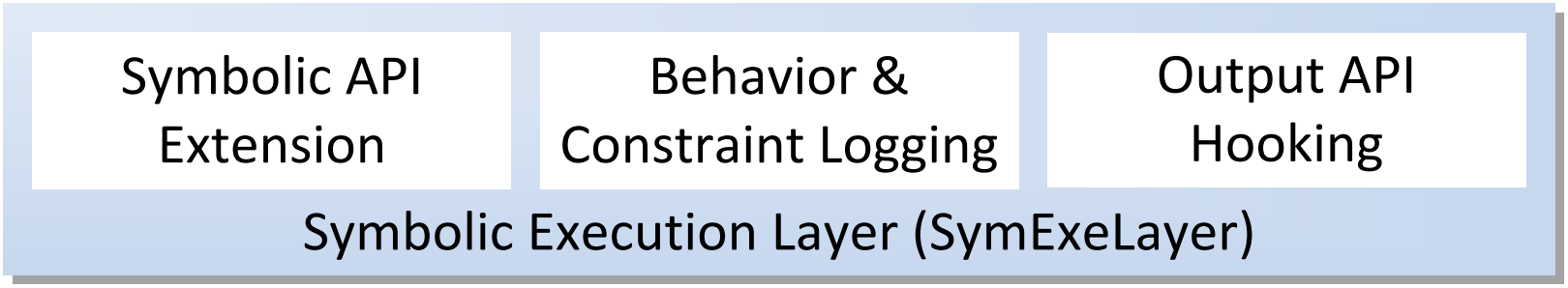
- Features
  - Require NO vulnerable application or testing environments
  - Expose attack behavior of each step under different conditions
  - Generate IDS signature just in minutes
  - Provide Day-One defense against new scripts

MetaSymploit Architecture



Computer Science
NC STATE UNIVERSITY

MetaSymploit Architecture
-Symbolic Execution Layer



- Symbolize APIs to return symbolic values
  - APIs depend on environment/target
  - APIs generate dynamic payload content
- Capture fine-grained attack behaviors and conditions
  - Behavioral APIs related to environment/target and payload
  - Branch constraints that reflect attack conditions
- Hook output API to capture the entire attack payload
  - The exact same payload received by target

Example of MetaSymploit

```
1 def exploit
2   connect()
3   preamble = "\x00\x4d\x00\x03\x00\x01"
4   version = probe_ver()
5   if version == 5
6     payload = prep_ark5()
7   else
8     payload = prep_ark4()
9   end
10  preamble << payload.length
11  sock.put(preamble) # Required by protocol
12  sock.get_once()
13  sock.put(payload) # Send attack payload
14  sock.get_once()
15  ... # vulnerability triggered
16 end
17 def prep_ark5()
18  payload = shellcode()
19  payload << rand_alpha(1167 - payload.length)
20  payload << "\xe98" + [-1172].pack("V")
21  payload << "\xeb\xf9"
22  payload << get_target_ret(5) # Tar_Ver: 5
23  payload << rand_alpha(4096 - payload.length)
24  return payload
25 end
```

Symbolic APIs:

```
probe_ver()
shellcode()
rand_alpha()
```

Behavior & Constraint Logging:

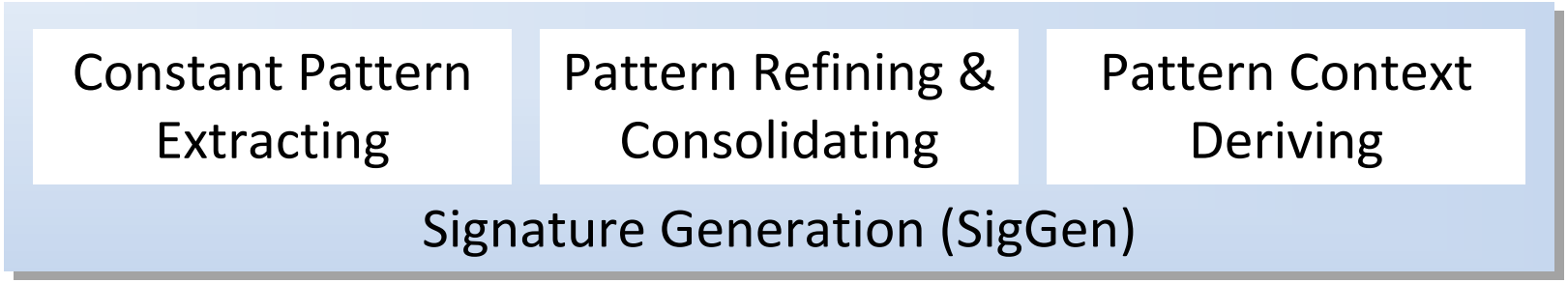
```
probe_ver()
sym_ver == 5
shellcode() &
get_target_ret()
```

Hook output API:

```
sock.put(payload)
```

Computer Science
NC STATE UNIVERSITY

MetaSymploit Architecture
-Signature Generator



- Extract signature patterns for specific attack payload
  - Based on network protocol format
  - Parse both symbolic and concrete contents
- Refine extracted patterns
  - Filter out benign/trivial patterns
  - Avoid duplicates based on pattern hashing
- Derive semantic context of patterns
  - Analyze the call sequence of behavioral APIs
  - AND all constraints as the overall attack condition

Example of IDS Signature

Line 23: payload => [<sym_shellcode, len=sym_integer>, <sym_rand_alpha, len=(1167-sym_integer)>, <"\xe9\x38\x6c\xfb\xff\xff\xeb\xf9\xad\x32\xaa\x71", 12>, <sym_rand_alpha, 2917>]

red is symbolic value, green is concrete value

```
alert tcp any any -> any 617 (
msg:"script: type77.rb (Win), target_version: 5, behavior: probe_version, stack_overflow, JMP to Shellcode with vulnerable_ret_addr";
content:"|e9 38 6c fb ff ff eb f9 ad 32 aa 71|";
pcre:"/[.]{1167}\xe9\x38\x6c\xfb\xff\xff\xeb\xf9\xad\x32\xaa\x71[a-zA-Z]{2917}/";
classtype:shellcode-detect; sid:5000656;)
```

Implementation

- Focus on Metasploit using Snort Rules
- Integrate into Metasploit Console
- Develop a lightweight symbolic execution engine for Ruby

Implementation

-Lightweight Symbolic Execution Engine for Ruby

- No modification to Ruby Interpreters
- Compatible with Ruby 1.8/1.9/2.0
- Leveraging Scripting Language Features
  - Debug tracing (`set_trace_func`)
  - Runtime context binding (Ruby's `Binding`)
  - Dynamic method overriding

Implementation

- Current Prototype:
  - Based on Metasploit 4.4
  - Ruby 1.9.3
  - Gecode/R & HAMPI as constraint solvers
  - Support 10 popular components in Metasploit
  - Cover 548 attack scripts

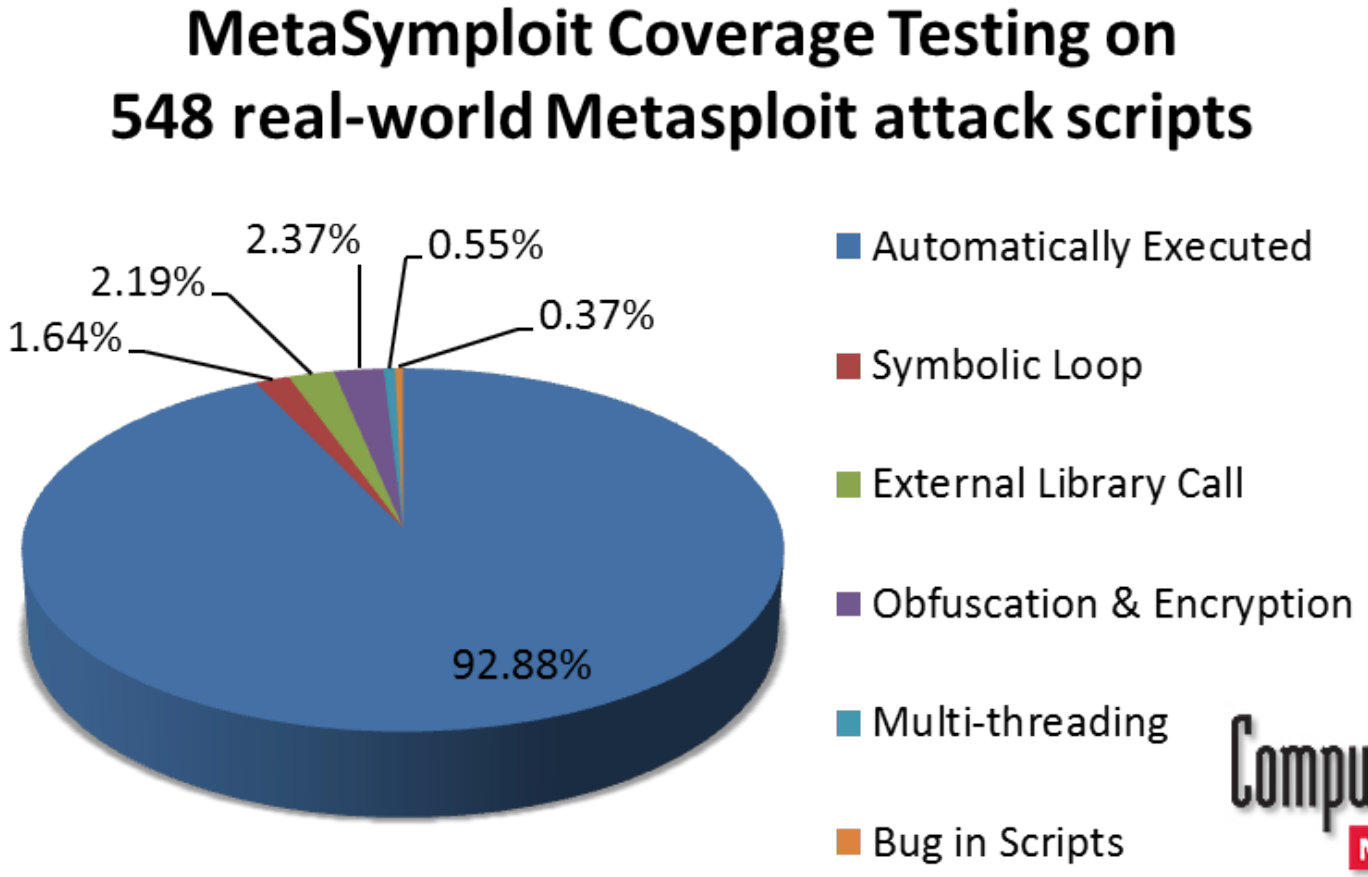
Evaluation

- Coverage Testing of Symbolic Execution Engine
- Effectiveness Validation using Real-World Metasploit Exploits
- Comparison with Official Snort Rules

Evaluation

-Coverage Testing

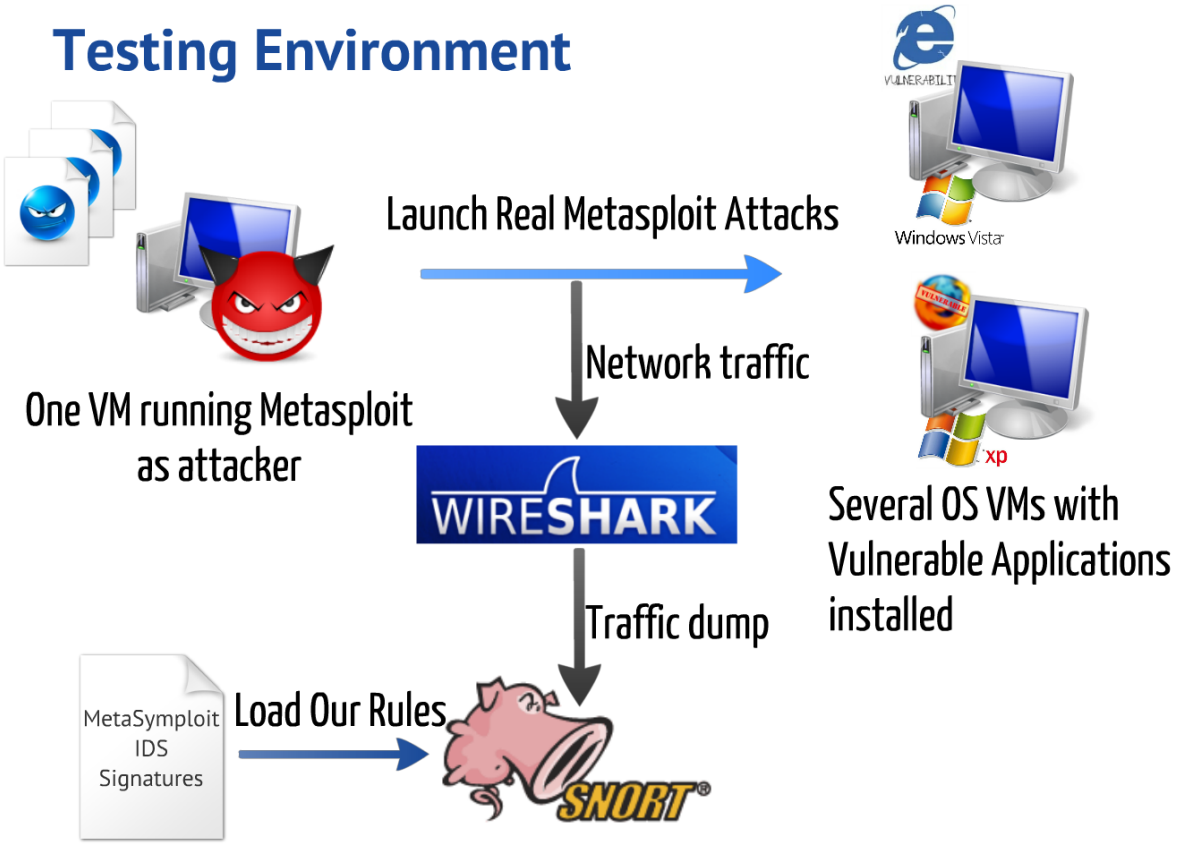
- Tested 548 attack scripts. Average < 1 minute per script
- 93% automatic, 4% manual effort, 3% not supported



Evaluation

-Effectiveness Validation using Metasploit

- Collect 45 Metasploit attack scripts targeting 45 vulnerable applications from exploit-db.com



Evaluation

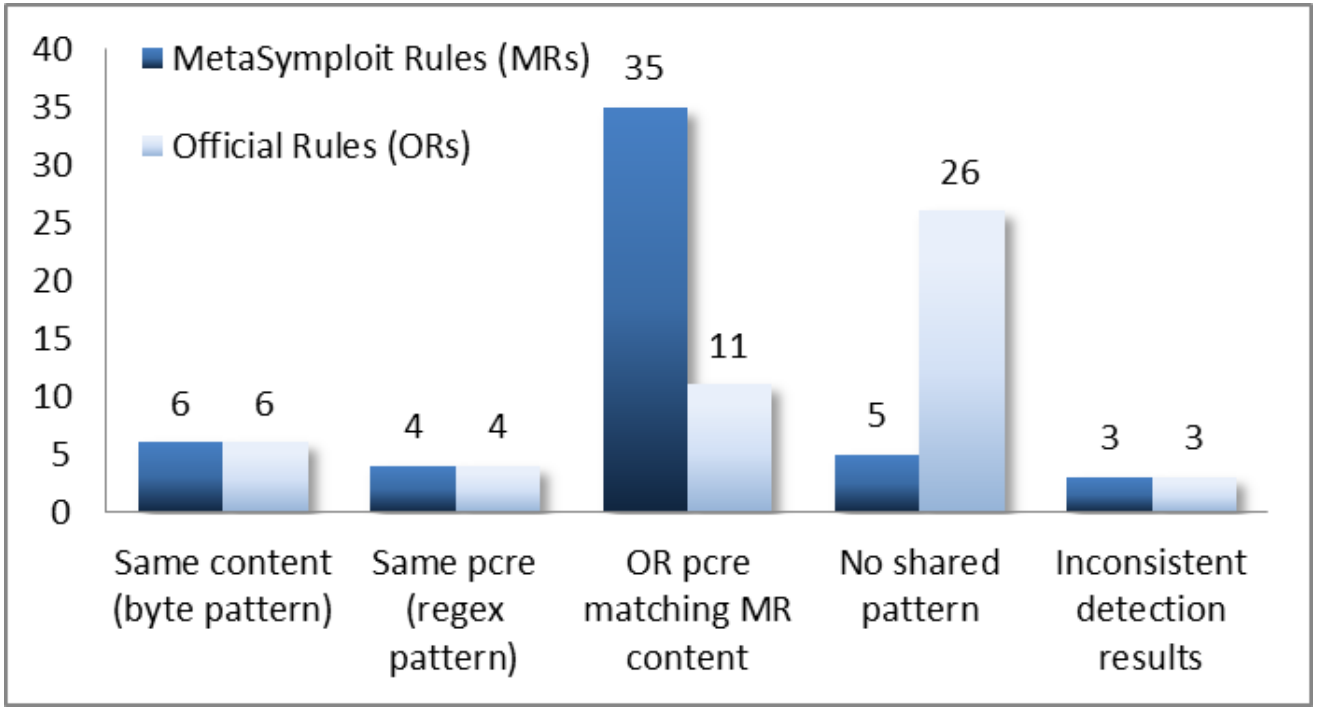
-Effectiveness Validation

- All attack payload packets are detected using MetaSymploit automatically generated Snort rules (100% true positive)
- Test with normal daily network traffic in CS labs for 2 months. No benign packet is mistakenly caught (0% false positive)
- The result is expected thanks to Pattern Refining in Signature Generator

Evaluation

-Comparison with official Snort Rules

- Compare with the official Snort rules (version 11/2012) for the previous 45 attack scripts.
- Only 22 out of 45 scripts have corresponding official Snort



Pattern comparison between 53 MetaSymploit generated rules and 50 official Snort rules for 22 Metasploit attack scripts

Computer Science
NC STATE UNIVERSITY

Evaluation

-Comparison with official Snort Signatures

- Updates
  - Version 07/2013 (snortrules-snapshot-2950)
  - The deficient rules are complemented with more rules to cover Metasploit exploits
  - Recent rules covers more public exploits, including **Meterpreter** shellcode
  - Introduce new rules: **`exploit-kit.rules`**, **`malware-tools.rules`**

Discussion

- The more attack scripts, the more MetaSymploit IDS signatures
  - Use as First Aid before patches are available
  - Use relevant sigs based on pattern context
- Limitations of classical symbolic execution
  - Infinite symbolic loop
  - Path explosion
  - Unsolvable constraints

Discussion

- Possible ways to bypass MetaSymploit
  - Develop script variants without releasing
  - Inject junk code/complex loops/non-linear constraints
  - Obfuscate script, like Blackhole Exploit Kit

Related Work

- Signature Generation
  - Attack Perspective:
    - Autograph [USENIX Security '04], Polygraph [S&P '05], Hamsa [S&P'06]
  - Vulnerability Perspective:
    - Vigilante [SOSP '05], ShieldGen [S&P '07], Bouncer [SOSP '07]
- Symbolic Execution for Security
  - Binary Level:
    - BitBlaze [ICISS '08], SAGE [NDSS '08], EXE [CCS '06], AEG [NDSS '11].
  - Scripting Languages focusing on web applications:
    - JavaScript [S&P '10], PHP [USENIX Security '06], Ruby on Rails [CCS '10]

Don't Get Me Wrong

- Metasploit is AWESOME! We like it!
- But public exploit should not be misused!

- When you publish a new exploit, attach IDS rule with it, to avoid bad guys taking advantage of your good contribution!

Thanks!

Questions?

Computer Science
NC STATE UNIVERSITY